A Bayesian model for updating fault type knowledge

LASSE HÜBINETTE[1] & JENS-PEDER EKROS[2]

[1]*Chalmers University of Technology*
Department of Total Quality Management
SE-412 96 GÖTEBORG, Sweden

[2]*Linköping University,*
Division of Quality Technology and Management
SE-581 83 LINKÖPING, Sweden
lasse.hubinette@mot.chalmers.se, jenek@ikp.liu.se

1. Introduction

During the past two decades, research in software quality has put much effort into the area of fault prediction. Despite these efforts, unsolved questions still remain, see particularly Fenton and Neil (1999a). Recently, a number of studies have suggested the use of Bayesian statistics for predicting faults (Fenton and Neil, 1999a; 1999b). So far, different types of faults have been analyzed as a group, using Bayesian statistics. In order to make better predictions, we suggest that faults be categorized and that Bayesian statistics be used to update our knowledge of the fault distribution between categories.

Most studies in the area of software fault prediction assume faults to be homogeneous in respect to their causes, for example Khoshgoftaar and Munson (1990), Compton and Withrow (1990) and Fenton and Neil (1999a). These studies treat faults as if they had the same characteristics and are caused by the same factors. The underlying assumption that faults are homogeneous is dangerous. If a novice programmer makes a careless mistake in the code and an experienced programmer makes a design error in a very complex part of the software, can the faults that arise from these mistakes be homogeneous? The answer is probably No. To assess the root causes of different faults, the faults may need to be categorized according to fault type.

Fault categorization is frequently used in areas other than software engineering. An example is Failure Mode, Effects and Criticality Analysis (FMECA), probably the most widely used and most effective analysis method for design reliability (O'Connor, 1995). Some researchers have suggested a categorization of faults into different fault types also in software engineering, e.g. Basili and Weiss (1984) and P1044 (1992). Categorization can then serve both as a tool to facilitate improvement and to improve the prediction model.

Apart from fault categorization, there are other problems in the area of software fault prediction. Fenton and Neil (1999a) introduced Bayesian belief networks in software fault prediction. A concept similar to Bayesian belief networks is influence diagrams. These enable reasoning under uncertainty and combine visual representation with a good mathematical basis. One important feature of influence diagrams is the possibility of allowing uncertainty about variables not ordinarily included in an analysis to influence the result, e.g. the type of tests performed, how difficult a problem is and what procedures were used. Such variables probably affect the number of faults to a high degree and in order to model these variables we can use both empirical evidence

and expert judgement if we utilize an influence diagram. The expert judgement becomes vital if we have only partial or subjective information about some of the variables.

In this paper, we present a model for updating knowledge about fault types in a coherent manner using Bayesian statistics. Similar approaches have been presented in different environments by authors such as Barlow (1998) and Sörensen Ringi (1995). The model is illustrated using an example from industry. It is important to note that this paper does not claim to present a complete fault prediction model. The results presented are intended to be tentative and hence conducive to further theory development.

In the next section, we will briefly explain the overall Bayesian approach of the model. The third section deals with the technical details. Finally, we compile the ideas by showing an application of the model from industry. In the two last parts, we also comment on various difficulties and constraints of the model.

The Bayesian Approach

The various steps in the Bayesian approach that is used can be seen in Figure 1.

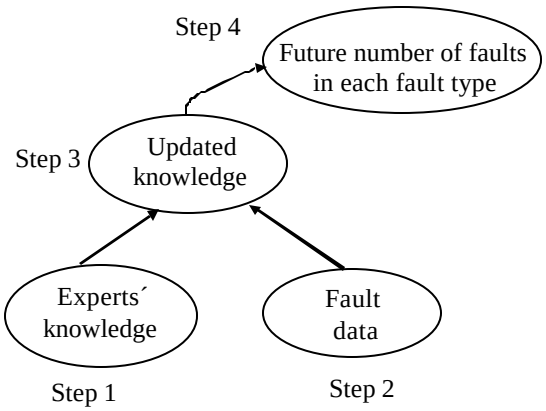

Figure 1 The different steps in the model

We have two different kinds of information, expert knowledge and empirical data. The experts' prior is (hopefully!) based on previous experience and all the differences between this project and the preceding projects. Both the experts' prior and the fault data are modeled using probability distributions. Instead of combining empirical data and expert knowledge through guesswork, Bayesian statistics is used. By introducing a formalized model, we achieve a consensus in our beliefs regarding the number of faults introduced instead of guessing the future number of faults. For further reading on Bayesian statistics and reliability, see e.g. Barlow (1998) and Crowder Kimber et al. (1991).

2. Statistical calculations

This section includes the calculations necessary for the model. Although we have tried to keep the technical details to a minimum, it may be hard for a practitioner or researcher with little experience in statistics to understand every part.

2.1. Model the knowledge

Software development consists of a large number of error possibilities, each giving rise to a fault with a small probability. Hence, by using a Poisson distribution, we acknowledge the special characteristic of the data, i.e. counts of events with small probabilities of the event. Software reliability models typically use such discrete distributions, see for example Musa (1998). The total number of faults is modeled as, $n \in Po(ls)$.

The parameter ls in the Poisson distribution consists of two parts. The first part, the intensity l, is distributed as the natural conjugate prior density for the Poisson distribution, namely the gamma distribution, $l \in \Gamma(a, b)$. The gamma distribution is a probability distribution rich enough to accommodate the prior knowledge. The gamma distribution also supports easy calculations since it is the natural conjugate prior density for the Poisson distribution.

In the model, a deterministic quantity is included termed s. The deterministic quantity s is an essential part of the model since it allows us to include the causes that make a system more fault-prone. If we establish that a small system has fewer faults than a large one, we can incorporate this knowledge into s, e.g. s can equal the lines of code (LOC) in the software. We can then predict faults/LOC instead of the number of faults. Instead of LOC, other metrics or combination of metrics can be used. Other possible metrics include complexity metrics, coupling metrics, difficulty of the problem, etc. In the end, it is the managers and developers who decide which sort of metric/metrics are to be used. One interesting use of the model arises when a new version of a software product is developed. Large-scale software systems are rarely built from scratch: instead, modifications and enhancements of existing systems are usually employed. Knowledge about product complexity and the quality of previous releases can be taken into account by using the deterministic quantity s. If the expert group concludes that it takes approximately 10 percent new work to complete the new version, this knowledge can be incorporated in s. If s was previously given the value 1, the new value for s in the new version should be 0.1.

How are the parameters a and b in the gamma distribution determined? The parameters a and b can be assessed by letting the experts express their belief in the intensity l. The intensity equals the number of faults if $s = 1$ and the fault density if $s = LOC$. The second step is to let the experts express their uncertainty about l. If the experts' belief is described with the negative binomial distribution, the constants a and b can be obtained as two of the parameters in the negative binomial distribution.

Typically, the experts, from past experience, have some idea of the proportional occurrence of faults between different fault types in a system. Fault types can naturally be modeled like a multinomial distribution since we are dealing with a number of categories (fault types) and observations in each category (faults in each fault type). To model the number of faults in each fault type, let k $(\mathrm{k} \geq 1)$ be the number of fault types and x_l $(l = 1, 2, ..., \mathrm{k})$ the number of faults classified in the category labeled l. Also, let the chance "probabilities" be $(y_1, y_2, ..., y_k) = \mathbf{y}$. The limiting proportion of faults y_l is referred to as the "chance" (or "risk") that a specified item will be defective. A chance, however, is not a person's probability but an unknown parameter of the binomial model based on the concept of a limiting proportion (which is not really observable) (Barlow, 1998).

Since the chance "probabilities" from the multinomial distribution are the parameters of interest, the natural conjugate prior density for these parameters is used, the dirichlet distribution. The dirichlet distribution is a rich enough probability distribution to accommodate prior knowledge, just as the gamma distribution was for the total number of faults. A change in the parameters

$q_1, q_2, ..., q_k$ in the dirichlet distribution corresponds to a change in the knowledge about the distribution of the faults across fault types. To express knowledge in the parameter **è** $(q_1, q_2, ..., q_k)$, the expert can first assign a percentage to the different fault types which indicates how common a certain fault type is in relation to another fault type. To express knowledge in absolute terms for obtaining the values of the parameter **è** , the expert can state how much confidence he/she has in the prediction in relation to empirical data. For example, if the a priori knowledge is worth the same as 100 counted faults in a real system, 100 is divided between the different parameters according to the percentage set earlier.

2.2. Updating knowledge

One way of visualizing the procedure of updating knowledge is by using an influence diagram. Influence diagrams are graphical representations of the relationships between random quantities which are judged relevant to a real problem (Barlow, 1998).

The model, visualized by an influence diagram, is developed as follows.

Step 1: The first step is to achieve a consensus about prior knowledge, i.e. the number of faults and the proportion of faults in each fault type for a certain system. As explained above, knowledge is modeled with the parameters a , b and the vector **è** $(q_1, q_2, ..., q_k)$. Two circles represent the deterministic quantity s. The parameters model the uncertainty concerning the fault distribution, i.e. l and the vector $\mathbf{y}$.

Step 2: The faults and the fault types are measured during a project (i). This data should have an effect on knowledge. Let the empirical data be denoted by n_i and the vector $\mathbf{x}_i$. Note that n_i equals the sum of the elements in the vector $\mathbf{x}_i$.

Step 3: Knowledge is revised in the light of the new data.

Step 4: The last step is to predict the future project (j), the total number of faults and the number of faults of a certain fault type. Let this future prediction be denoted by n_j and the vector $\mathbf{x}_j$.

The influence diagram shows the relationship between our random and deterministic quantities, Figure 2. The arcs correspond to possible conditional dependence.

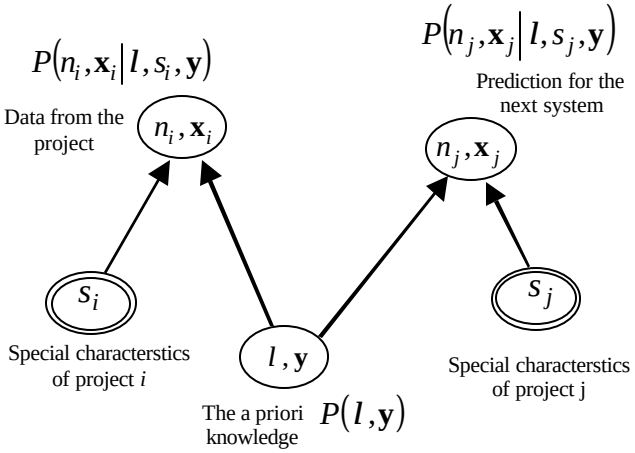

Figure 2 Influence diagram for project i and j

By using the laws of probability, we can update knowledge about fault types and then predict the fault content of the next system. We assume that l and $\mathbf{y}$ may be treated as independent, i.e. knowledge of one of these quantities does not affect knowledge of the other. This assessment of prior information is quite plausible from our point of view. The total number of faults may vary without changing the proportional occurrence of faults in fault types. Naturally, there may be differences in the proportional occurrence of faults between different fault types in one system compared to another. The reason is quite simple: different fault types have different root causes. However, the model provided is a first step in understanding and modeling the way in which faults vary across fault types. Therefore, it seems reasonable to simplify the model by assuming l and $\mathbf{y}$ as independent, although further enhancements of the model should explore the justification of such an assumption.

To update the information about l and $\mathbf{y}$, i.e. to calculate the probability of l and $\mathbf{y}$ given the new data n_i and $\mathbf{x}_i$ ($P(l, \mathbf{y} | n_i, \mathbf{x}_i)$), we need to reverse the arc between the data from the project and the a priori information. In order to update knowledge, i.e. reverse the arc, Bayes' theorem is applied.

Since we have used the natural conjugate distributions, the Bayesian updating algorithm provides posterior distributions that are still within the gamma and dirichlet families respectively. This means that we do not have to calculate the probabilities using Bayes' theorem whenever we use the model. The parameters are simply updated as:

$$a \rightarrow a + n$$

$$b \rightarrow b + 1$$

$$q_l \rightarrow q_l + n_l$$

Since our prior and posterior probability distributions are parameterized in the same way, the predictive distributions of faults and fault types are calculated in the same way regardless of whether we only have prior information or observed data from the system. To predict the future number of faults for a new, similar system, the negative binomial distribution is used. The prediction is obtained by using the parameters a and b from the gamma posterior distribution in the negative binomial distribution. It is more difficult to predict the number of faults in the different fault types. However, the posterior dirichlet distribution provides the expected number of faults in each fault type for a new system. This should in most instances be sufficient and only under special circumstances should a complete prediction be necessary. One such case arises when resource allocation to testing and inspection techniques based on fault type is important. In such a case, the robustness of the expected number of faults in one fault type could be useful. In order to assess the robustness, simulations of the so-called Dirichlet-Multinomial Gamma-Poisson model could be performed. For further information, see Sörensen Ringi (1995).

3. Application

The empirical data used to illustrate the model are gathered from a study at Saab Aerospace. The application is intended to demonstrate how knowledge about the number of faults in each fault type can be updated.

3.1. The system

The system investigated is a subsystem, Built In Test (BIT), of the Electrical Flight Control System (EFCS) in JAS 39 Gripen, a fighter aircraft developed and built by Saab Aerospace. Its main objective is to ensure, prior to each flight, that the state of the EFCS hardware is such that the flight can be performed safely.

During the development of BIT, Saab Aerospace followed the guidelines in the DOD-STD-2167A standard, which is further described in the (U.S. Department of Defense, 1988). The program is divided into 256 modules. It is a complex program, resulting in nearly 1500 faults to date. The code was programmed mostly using ADA (204 modules), which was the standard programming language. We have analyzed only the faults in the ADA modules (1112 faults) in order to avoid the complication of more than one programming language becoming an explanatory factor.

3.2. Using the model

The following section describes a brief examination of the application at Saab Aerospace. This section is intended only to provide an idea of how the model could be applied and should not be taken as state-of-the-art practice. The model needs to be tailored more or less from organization to organization.

No detailed explanation of the different fault types used is provided. The reason is simple. Fault types must be customized to a company's own organization to fully benefit from the categorization. The definitions of the fault types are meaningful to the organization that collected and analyzed the data, but may not be suitable in other environments

3.2.1. Step 1

The constant s was chosen to describe how demanding the system was to develop. Since no other system was under development, s was given the value 1.

Note that the a priori information was collected after the system was developed. This is of course not the correct procedure to employ, but was the only choice left since the system had already been developed. However, the members of the expert group were not aware of the actual result in terms of the number of faults across fault types.

Cooke (1991) states that using the Bayesian approach for multiple experts is rather tentative and better solutions may be anticipated. It was therefore decided that only one developer and one specialist from Saab Aerospace should be included in the expert group. It is probably more important to make use of persons who are experts in their field instead of having many persons in the expert group. However, this is a choice of those who apply the model.

First, the expert knowledge needs to be modeled. As was explained earlier, the gamma and dirichlet distribution distributions are used to model knowledge. Instead of forcing the experts to assign figures to the parameters, the following approach was used. First, the experts had to predict the number of faults in the software, i.e. specify their belief in a specific number (l). The second step was to let them express their uncertainty about these figures (uncertainty about l). After a number of different approaches, it proved easiest to show them distributions expressing various levels of uncertainty. If they stated that they were fairly certain about the result, we tried to express this by showing them a distribution concentrated around their prediction. If, on the other hand, they reported large uncertainty we used a distribution with large variation. The whole procedure was conducted for each member of the expert group, after which they had to reach agreement. They

discussed their numbers and graphs and explained their different assumptions behind their predictions. The model of their knowledge can be seen as the a priori distribution in Figure 3.

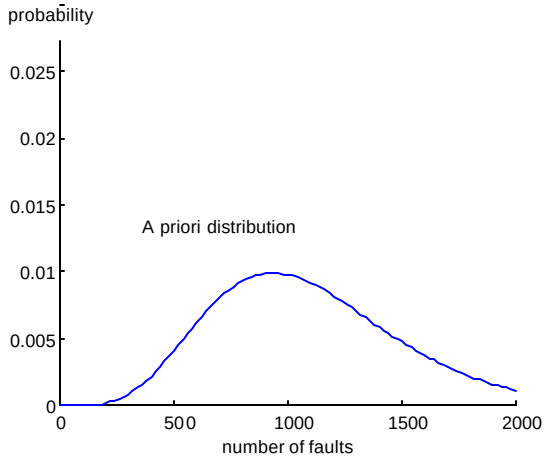

Figure 3 The prior distribution of the total number of faults

To attain the experts' knowledge regarding the fault types the following procedure was used. First they had to set a percentage on the different fault types which indicated how common a certain fault type was in relation to another fault type. To express their knowledge in absolute terms, in order to obtain the values for the parameters $(q_1, q_2, \ldots, q_k)$ in the dirichlet distribution, they had to state how much confidence they had in their prediction in relation to empirical data. If they felt that the a priori knowledge was worth as much as 500 counted faults in a real system, 500 was divided between the different parameters according to the percentage set earlier. The results can be seen in Table 1.

To establish a good prior, a preceding system was analyzed. The experts had to take into account all the differences between this project and the preceding project. Examples of issues that need to be considered are the type of tests used, the processes and procedures used and the complexity of the systems.

The procedure used, in which experts discussed their different beliefs, improved the understanding of the software development process. Instead of guessing the future number of faults, a consensus was achieved regarding belief about the number of faults introduced. This discussion also indicates what the experts believe are the reasons behind the faults, which can be assessed by performing studies of the desired causes.

3.2.2. Step 2

From the development of the system, information about the number of faults was obtained. The result can be seen in Table 1. The faults were distributed across seven fault types and the total number of faults found was 1112. The experts' belief corresponds to the parameters **è** and the actual distribution corresponds to the vector **x**.

Fault type	Experts' belief	Actual distribution	Updated belief
Computation	5%	3%	4%
Data handling	20%	26%	23%
Data problem	5%	6%	5%
Document quality	20%	12%	17%
Documentation	20%	22%	21%
Interface/timing	10%	22%	15%
Logic	20%	9%	15%
Total	100%	100%	100%

Table 1 Information about the developed system

3.2.3. Step 3

Both the information about the total number of faults and its variation, and the distribution of faults across the fault types, was updated by combining the experts' prior knowledge with the empirical results. Figure 4 shows the updated distribution. It can be seen that the experts' belief was in good agreement with the actual number of faults. However, some uncertainly about the outcome was expressed by the experts. The result of the model was a more confident belief with smaller uncertainty about the number of faults in a future system of this type.

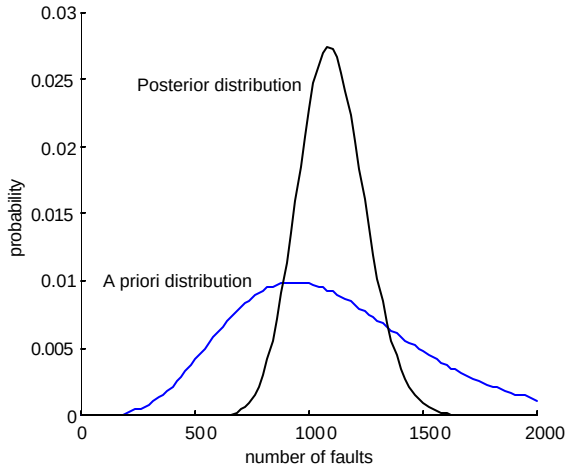

Figure 4 The prior and posterior distribution of the total number of faults

Knowledge about the distribution across fault types was also updated by combining expert knowledge with empirical results. See Table 1 for the results.

Potential short-term benefits are better inspection and testing processes. These can be improved by learning what types of faults are made and what types of faults are absent or missed in different phases. One suggested improvement is to introduce training activities in creating better design documents, which should focus on avoiding problems in the fault types data handling, documentation, document quality and timing/interface. The faults introduced in the coding phase were mainly data handling faults, which can be dealt with by creating a list of common mistakes of data handling faults that is fed back to each programmer. This makes the programmer more aware of the problem so that he becomes more careful about making this sort of mistakes. A fault type distribution profile can be said to serve as a measurement tool for helping help both management and technical personnel isolate faults earlier in the development phases.

A large number of interface/timing faults implies that a problem exists with the synchronization between the modules and between the modules and the hardware. The close interaction in BIT between the hardware and software may be the explanation. BIT was programmed to use three channels, which means that the execution of the program is simultaneously executed in three different systems (three processors, three memories, etc.). The use of three channels requires the programmer to have detailed system knowledge. This was not recognized before the systems were developed, as is evident from the experts' belief of approximately 100 faults in the interface/timing fault type. The actual result was in the region of 250 faults and since these faults were quite expensive, in terms of correction, activities to reduce the number of interface/timing faults should be introduced before the next system is developed. Certain training activities may be introduced to give the programmer more knowledge about the surrounding environment and the use of three channels. The composition of the team that developed BIT is also highly relevant when discussing system knowledge. Choosing consultants with prior experience from projects similar to BIT would probably not only improve their own performance but also allow the other project members to better understand the problems. If the model had not been used, too much attention would have been directed towards logic faults instead of towards the interface/timing faults in the next system developed.

A prediction of the number of faults introduced provides useful help in deciding when to release the software to the customer. Such a prediction helps both the organization and customers to make reliable decisions concerning the implementation date of the software.

4. Concluding remarks

To assess the root causes of faults, we may need to categorize the faults according to fault type. The reason is simple: different causes affect certain fault types in a specific way. Categorization can serve as a tool both to facilitate improvement by assessing the causes and to improve the prediction model.

We have identified that one important task is to combine expert knowledge about fault types with empirical fault data from previous and current projects. Expert judgement becomes vital if we only have partial or subjective information about some of the important variables. In this paper, a Bayesian approach to the problem is proposed. When using the method, knowledge about fault types increases and can be used to analyze the root cause of each fault type respectively.

The ability to use influence diagrams to predict faults depends on the regularity and stability of the development process. Organizations that do not collect metrics or follow the prescribed development process cannot hope to apply influence diagrams in their organizations successfully.

As was mentioned before, the model is intended to be an intermediate step and provide an opportunity for refinement and reformulation. The treatment of the issues discussed in this work is

only a first step; further research is certainly needed before the method gains any real practical importance. One improvement area is undoubtedly the way in which expert knowledge is obtained.

Acknowledgements

The authors wish to thank Bo Bergman and Mats Lörstad for helpful comments on this paper. This work was supported by the Swedish National Board for Industrial and Technical Development (NUTEK), administered by the Swedish Institute for Applied Mathematics (ITM). The authors would especially like to thank Saab Aerospace for their contribution to the study.

References

Barlow, R. E. (1998). *Engineering Reliability*. Philadelphia, American Statistical Association and the Society for Industrial and Applied Mathematics.

Basili, V. R. and D. Weiss (1984). "A methodology for collecting valid software engineering data." *IEEE Transactions on Software Engineering* 10: 728-738.

Bernardo, J. M. and A. F. M. Smith (1994). *Bayesian Theory*. New York, John Wiley & Sons.

Compton, B. T. and C. Withrow (1990). "Prediction and Control of ADA Software Defects." *The Journal of Systems and Software*: 199-207.

Cooke, R. M. (1991). *Experts in Uncertainty*. New York, Oxford University Press.

Crowder, M., A. Kimber, et al. (1991). *Statistical analysis of reliability data*. London, Chapman & Hall.

Fenton, N. and M. Neil (1999a). "A Critique of Software Defect Prediction Models." *IEEE Transaction on Software Engineering* 25: 675-689.

Fenton, N. and M. Neil (1999b). "Software Metrics and Risk." *FESMA 99, 2nd European Software Measurement Conference*: 39-55.

Khoshgoftaar, T. M. and J. C. Munson (1990). "Predicting Software Development Errors Using Complexity Metrics." *IEEE Journal of Selected Areas in Communications* 8: 253-261.

Musa, J. D. (1998). *Software Reliability Engineering: More Reliable Software Faster Development and Testing*. New York, Computing McGraw-Hill.

O'Connor, P. D. T. (1995). *Practical Reliability Engineering*. Chichester, John Wiley & Sons.

Ohlsson, N. (1998). Towards effective fault prevention - An Empirical Study in Software Engineering. Department of Computer and Information Science. Linköping, Linköping University.

P1044, I. (1992). A Standard Classification for Software Anomalies (draft), *IEEE Computer Society Press.*

Sörensen Ringi, M. (1995). On Bayesian System Reliability Analysis. Division of Quality Technology, Department of Mechanical Engineering. Linköping, Linköping University.

U.S. Department of Defense (1988). DOD-STD-2167A. Washington, D.C., Defense System Software Development.